Accelerated Foveated Rendering based on Adaptive Tessellation

Tiwary, Ankur; Ramanathan, Muthuganapathy; Kosinka, Jiri

Published in:
Eurographics 2020 - Short Papers

DOI:
10.2312/egs.20201003

Accelerated Foveated Rendering based on Adaptive Tessellation

A. Tiwary[1], M. Ramanathan[1], and J. Kosinka[2]

[1]Department of Engineering Design, Indian Institute of Technology Madras, India
[2]Bernoulli Institute, University of Groningen, Netherlands

Abstract

We propose an optimization method for adaptive geometric tessellation, involving the saccadic motion of the human eye and foveated rendering. Increased demands on computational resources, especially in the field of head-mounted devices with gaze contingency make optimization schemes pertinent for a seamless user experience. For implementing foveated rendering, our algorithm tessellates a 3D model in real-time based on the location of the user's gaze, substituted with a mouse cursor in this project as a proof of concept. Saccades and fixations of the human eye are simulated by delaying the process of tessellation and rendering by the minimum time taken to complete a saccade. Calculations required for tessellation and rendering the changes on the screen are stalled as and when the eye fixates after a saccade. The paper walks through our contribution by describing the theory, the application method, and results from our user study evaluating our method.

CCS Concepts

• ***Computing methodologies*** → ***Rendering; Mesh geometry models;*** *Graphics systems and interfaces;* • ***Human-centered computing*** → *Virtual reality;*

1. Introduction

Evolved over millennia, the Human Visual System (HVS) is apt for viewing the world as we know it. Computers and display devices have seen an upsurge in sophistication and the quality of graphics ever since their conception. With limited processing power available, more performance requires optimization techniques. Foveated rendering is one such scheme, adapted from nature.

Foveated rendering is a technique that leverages the rapidly declining perceived quality of human vision towards the periphery in order to speed up 3D rendering [PSK*16]. Differential allocation of resources occurs across the display, with the image quality degrading on moving away from the gaze location. As the HVS tends to ignore the finer details towards the periphery, the user does not realize the degraded resolution. The optimization can be introduced at different stages in the rendering pipeline. Our work varies the amount of smooth tessellation done to the 3D mesh of a geometric model to get a finer output locally.

The human eye performs various distinct movements. Saccades are rapid eye movements designed to shift the fovea to objects of visual interest [TTRF15]. A reduction of visual sensitivity occurs around the time of saccades, to maintain perceptual stability [BKHK09]. Algorithms manipulating and exploiting saccadic suppression for optimization are topical areas of research. Here, we discuss its adaptation into geometric tessellation with foveated rendering; see Figure 1.

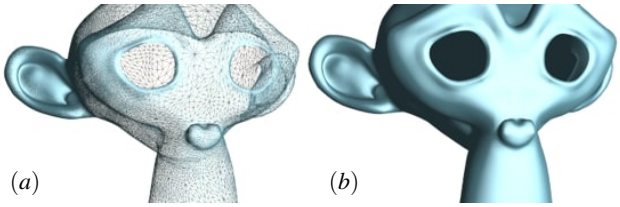

Figure 1: *Output of hardware tessellation. (a) A wireframe model (based on QAS applied to the Blender Suzanne model) with the cursor placed on the ear of the model, showing adaptive tessellation in action. (b) The viewer sees the 3D model smoothly shaded.*

An immediate benefit from this work is realised in computationally expensive applications such as head-mounted virtual/augmented reality (VR/AR) devices with gaze contingency. The proof of concept has been provided by an application specifically created for the purpose. The tessellation method followed by this application is Quadratic Approximation of Subdivision Surfaces (QAS) [BS07], but the concept can be extended to all surface schemes based on geometric patches, such as [LSNCn09]. A controlled user study gave us insights into the potential of our method.

This paper is divided into five parts. After this introduction, Section 2 introduces the involved concepts. Section 3 details the implementation procedure. The experimental procedure and the corresponding results are covered in Section 4. Finally, Section 5 summarises the results and proposes future work.

2. Related Work and Background

We start by briefly recalling related work and relevant concepts.

2.1. Previous Work

Foveated rendering by manipulating peripheral contrast and shading coarseness was covered in [PSK*16]. [Lin16] describes an implementation of adaptive tessellation on a 3D object using hardware resources, and then applies blur to simulate depth perception. [KSL*19] describe a novel neural reconstruction for images after a stochastic dropping of peripheral pixels from a sparse input video stream. A foveated rendering algorithm by varying the refresh rate of sections of the display was devised in [GFD*12]. The screen was divided into discrete layers (2 or 3 as per need) of sampling rates arranged in concentric circles about the gaze location. The inner circle had the highest sampling rate, which dropped in the outer layers. An improvement in the average performance by a factor of 6 was observed. [SIGK*16] evaluated several possible ways of implementing foveated rendering, using metrics required for various designing aspects, such as the foveal window size and peripheral resolution. Adaptive tessellation is one of the four methods evaluated. Subdivision and tessellation are very popular methods for smoothing a carefully designed coarse mesh into a high-resolution smooth version using a set of local rules, typically on the GPU [BS07, LSNCn09]. Apart from QAS, other tessellation schemes can also be followed, with minor modifications. The principle, as described in Section 3, however, remains the same.

2.2. Foveated Rendering

The human eye has photo-receptors called cones and rods. The density of cones near the center of the eye is much higher than the density near the periphery. The fovea centralis is the region with the highest cone concentration. The result is a small viewing cone where the sensitivity is maximal, with progressive degradation along either axis; see [WB01, Figure 2]. This is exactly what foveated rendering takes advantage of.

2.3. Saccades and Fixations

Saccades are rapid eye movements which align the fovea with the region of interest for the best resolution of spatial details [ATM*17]. Saccadic suppression is the phenomenon where the sensory information is significantly reduced. This gives a person a stable depiction of the surroundings during a saccade [BKHK09]. A saccade generally lasts from anywhere between 20 to 200 milliseconds (ms) [ATM*17], with a few tens of milliseconds going into restoring complete sensitivity. The fastest time taken for an entire saccade and fixation to happen is 20ms and 100ms, respectively [PHR*08]. For foveated rendering, calculations for tessellation occur at the screen refresh rate, regardless of whether the cursor is held still or is in motion.

Our solution is to efficiently use (re)calculations only where (foveated rendering) and when (saccades) needed. For a display working at 120Hz, each update occurs once every 8.33ms. Our proposed solution only performs the calculations when the cursor is in motion and once every 20 milliseconds, reducing the number of calculations by a minimum of 14 times. The performance increase is even higher for displays with higher frame rates.

3. Method and Implementation

Our method has been developed in C++ with Qt for control and OpenGL (with tessellation shaders) for rendering. This section defines the step-by-step methodology followed to implement a saccade and fixation based tool for foveated rendering, the underlying concepts of which were discussed in Section 2.

- Identifying the gaze location of the user is the primary task (the cursor location in our implementation). The coordinates of the cursor location are transformed from the view-port space coordinates to normalized device coordinates. Scaling is then applied to transform the coordinates to the four dimensional homogeneous clip space. The coordinates are then 'unprojected' to the camera space coordinates by multiplying the 4-dimensional vector with the inverse of the projection matrix.
- In the camera space, a ray is cast from the camera to the cursor location, and it is normalized to obtain the unit vector in its direction (the z-coordinate is on either the near or far clip plane). This ray is passed to the tessellation control shader.
- The perpendicular distance of each (transformed) vertex from the ray is measured. Tessellation levels for each edge of the triangular QAS patch are assigned based on the inverse of the ray-vertex distance, multiplied with a suitable scaling factor. The scaling factor is adjusted to allow the region of high resolution to be greater than the foveal cones base. We optimised this parameter through user experiments.
- Once the cursor is in motion, we store its position from the current and previous frame. The time taken for performing the operations in each iteration is stored. The speed of the cursor is calculated by dividing the length between the old and new cursor location by the time taken. Predicting the location at which the next possible fixation will occur is done by multiplying the cursor speed with the unit vector in this direction and then with the minimum time required for a saccade.
- The set of calculations in our setting refers to: finding the possible fixation location; creating a ray through a given cursor location and transforming it from the viewport space to the camera space; and performing subdivision in the tessellation shader for each polygon (triangle in the QAS case). This set of calculations is performed once and the function is not called for the next 20ms. During this time the only update that occurs is the predicted location of next fixation.
- If the cursor keeps moving after 20ms, the process is repeated. In case a fixation occurs, no calculations are done till the next saccade is initiated. In case a fixation occurs in-between the 20ms steps, the scaling factor ensures that the error in gaze location prediction is imperceptible. Future work can vary the scaling factor dynamically based on parameters such as cursor velocity or user distance from the screen.

4. User Study

A study with 15 participants was performed to evaluate the proposed method. A successful method and implementation has been achieved when the participants are not able to distinguish between

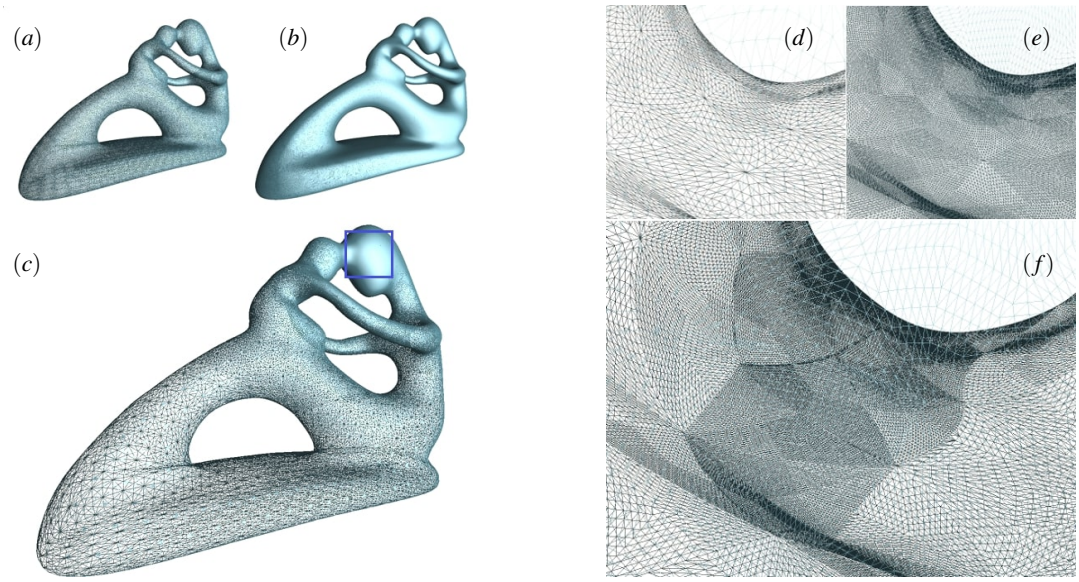

Figure 2: *(a) A normally tessellated 3D model with approx. 100K triangles. (b) A normally tessellated 3D model with approx. 1M triangles. (c) A foveated model with varying tessellation density. The blue box refers to the gaze point. All three images show wireframe models for clarity. On the right, (d–f) show insets of the three versions (a–c), respectively. Observe the adaptive nature of tessellation in (f).*

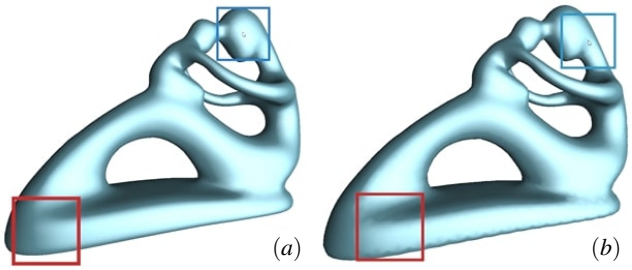

Figure 3: *A still from the video with regular (uniform) tessellation (a) and a still with the saccade fixation implementation (b).*

regular tessellation and our adaptive algorithm. As the absence of a gaze tracker is clearly a major concern for the evaluation of our cursor-based implementation, the experiment was designed around this limitation. A viewing square of appropriate size was created. The user was asked to confine their visual attention within this viewing box, which moved across the display screen with the cursor. Each participant was first prepared for the task of tracking the viewing box. A video with dots appearing at random was used for this purpose. The user was required to direct their gaze to each new dot on the screen as it appeared. This exercise was repeated until the user was comfortable with tracking dots effortlessly, i.e., without considerable diversion of attention. In order to facilitate the study and enable the participants to easily answer the questions which were to follow, the effect of tessellating a surface was explained. The difference between a coarse and finely tessellated 3D model was consequently shown to the user.

A set of three videos was prepared for the main evaluation (see supplementary material). The first video had the viewing box moving on the screen with a model which had been tessellated to a million polygons (illustrated in Figure 2(b, e)). No changes actually occurred during the run-time of this video; this acted as our baseline. The second video had the foveated rendering algorithm running in the background. The saccades and fixations acceleration was employed in the third video. The responses for the first video were used to relatively evaluate the performance of the rest.

The viewing box performed two different motions. The first motion was a continuous clockwise movement around the model. The second movement made the viewing box appear and disappear at certain intervals of time on the same trajectory. This simulated a possible path a human eye might use to view the object. The subjects went through three iterations of each video in a randomised order to prevent bias and maintain consistency. Responses to the questions asked were recorded. Static images (illustrated in Figure 3) were shown afterwards. On these, the participants reported similarity between the two viewing box modes on a scale of 1 to 10, with 1 meaning uniquely distinct and 10 meaning no difference.

The viewing box can be a cause for distraction. Paintings, photographs and most displays on various devices around us are generally rectangular. A square box of side length two centimeters was thus chosen for the task, as it should be more convenient to 'look into'. The participants were seated comfortably with the display and chair oriented ergonomically. The screen was placed 60cm from the participant's eyes, with an error margin of 5cm. At this distance, the sharpest field of vision corresponds to a span of 1.33cm on the display surface. Errors due to shifting of gaze from the centre of the viewing box and those due to micro-saccades were considered, resulting in the given final form of the viewing box. A backlit LED display with a resolution of 1980x1080 at a refresh rate of 60Hz was used for the experiments.

The amount of tessellation was measured in terms of the number of generated triangles. For example, when a coarse model with a few hundred polygons was converted to one with a million poly-

gons, the density of the latter has been used as a reference scale for comparison. For the videos, the distribution of mesh density was adjusted so that there existed a million polygon model density at the gaze location. The density at the periphery had the equivalent of a 20K-triangle model. The distribution of tessellation was controlled to limit the maximum number of triangles to be less than 8 times a fully tessellated model. The number of calculations thus reduces by a minimum of the same factor. Therefore, theoretically there was at least a 112 times reduction in the number of calculations compared to a standard tessellation method.
Participants selected for the experiment had no prior knowledge of foveated rendering. For both motions of the viewing box across the screen, no inadequacies in tracking ability were reported. Tables 1 and 2 report the results of the tests.

Table 1: *Results for each iteration per motion and video. The reported values signify the number of successful trials where no visual distractions were reported.*

Motion	Video 1			Video 2			Video 3		
Iteration	I	II	III	I	II	III	I	II	III
First	14	14	15	14	15	15	15	15	15
Second	13	15	15	14	15	15	14	15	15

Table 2: *Means and modes of the values reported by participants regarding the similarity inside the viewing boxes (scale 1 to 10).*

Average Value	Video 1	Video 2	Video 3
Mean	8.26	4.13	3.73
Mode	8	3	4

It was observed that the majority of users did not spot any discerning occurrence on the screen. The few discrepancies were recorded during the first few trials for a few users, which can be attributed to the experimentation method, rather than the algorithm itself. This is substantiated by the fact that the issues occurred more in the first (baseline) video, where we know no changes occurred in the model. When asked to compare the image quality between the two boxes (illustrated in Figure 3), all participants reported a sharp difference between the image quality for the proposed algorithms, and minimal difference for the regular tessellation case, as expected. Participants were then asked to observe the actual changes that occur on the screen as the algorithms functioned by pointing them out. None of the subjects reported having observed any such change during the experiment.

5. Conclusion

We have observed that users were not able to distinguish between the functioning of the foveated rendering algorithms and regular tessellation. Unwanted artifacts and visual distractions were reported in around 2.5% of the total cases. Statistically, around 6% of the trials showed some visual pops during the first set of trials, which went down to 1% by the second set of trials. No pops of any kind were reported during the final set of trials. The algorithm can theoretically give a much better performance gain, judging by the number of calculations at every update, if used on larger scenes with more models/polygons, and on devices with an even higher frame rate.

We are now implementing this algorithm using an eye tracker as a natural continuation. New problems arise in that case, primarily due to latency, hardware dynamics and accuracy. Fusion with other adaptive tessellation schemes, such as those based on the proximity and orientation of the model to the camera in the scene, the curvature of model sections, are being explored. Better prediction models for fixations sites and the incorporation of other eye movements is required to build a robust foveated rendering algorithm.

Acknowledgements

Our implementation builds on the QAS implementation by Jens van der Meer and Tim Oosterhuis, which in turn builds on an OpenGL/Qt framework provided by Pieter Barendrecht. This research is based on the first author's internship at the University of Groningen.

References

[ATM*17] ARABADZHIYSKA E., TURSUN O. T., MYSZKOWSKI K., SEIDEL H.-P., DIDYK P.: Saccade landing position prediction for gaze-contingent rendering. *ACM ToG 36*, 4 (2017). 2

[BKHK09] BREMMER F., KUBISCHIK M., HOFFMANN K.-P., KREKELBERG B.: Neural dynamics of saccadic suppression. *Journal of Neuroscience 29*, 40 (2009), 12374–12383. 1, 2

[BS07] BOUBEKEUR T., SCHLICK C.: QAS: Real-time quadratic approximation of subdivision surfaces. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), PG '07, IEEE Computer Society, pp. 453–456. 1, 2

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3d graphics. *ACM ToG 31*, 6 (Nov. 2012), 164:1–164:10. 2

[KSL*19] KAPLANYAN A. S., SOCHENOV A., LEIMKÜHLER T., OKUNEV M., GOODALL T., RUFO G.: Deepfovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Trans. Graph. 38*, 6 (Nov. 2019). 2

[Lin16] LINDEBERG T.: *Concealing rendering simplifications using gazecontingent depth of field*. Master's thesis, KTH, School of Computer Science and Communication (CSC), 2016. 2

[LSNCn09] LOOP C., SCHAEFER S., NI T., CASTAÑO I.: Approximating subdivision surfaces with Gregory patches for hardware tessellation. *ACM ToG 28*, 5 (Dec. 2009), 151:1–151:9. 1, 2

[PHR*08] PANNASCH S., HELMERT J. R., ROTH K., HERBOLD A.-K., WALTER H.: Visual fixation durations and saccade amplitudes: Shifting relationship in a variety of conditions. *Journal of Eye Movement Research 2*, 2 (Dec. 2008). 2

[PSK*16] PATNEY A., SALVI M., KIM J., KAPLANYAN A., WYMAN C., BENTY N., LUEBKE D., LEFOHN A.: Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph. 35*, 6 (Nov. 2016). 1, 2

[SIGK*16] SWAFFORD N. T., IGLESIAS-GUITIAN J. A., KONIARIS C., MOON B., COSKER D., MITCHELL K.: User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception* (New York, NY, USA, 2016), SAP '16, ACM, pp. 7–14. 2

[TTRF15] TERMSARASAB P., THAMMONGKOLCHAI T., RUCKER J. C., FRUCHT S. J.: The diagnostic value of saccades in movement disorder patients: a practical guide and review. *Journal of Clinical Movement Disorders 2*, 1 (Oct 2015), 14. 1

[WB01] WANG Z., BOVIK A. C.: Embedded foveation image coding. *IEEE Transactions on Image Processing 10*, 10 (2001), 1397. 2